UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/687,233	10/15/2003	Martin Runte	6570P015	8107

45062 7590 06/04/2007

SAP/BLAKELY
12400 WILSHIRE BOULEVARD, SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
06/04/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/687,233	RUNTE ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 May 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on May 17, 2007.

2. **Claims 1-35** are pending.

3. **Claims 1, 2, 6, 8-12, 23, 26-28, and 32-35** have been amended.

4. **Claims 36-38** have been cancelled.

5. The objection to the abstract is withdrawn in view of Applicant's amendments to the abstract.

6. The objections to the specification are withdrawn in view of Applicant's amendments to the specification.

7. The objections to Claims 1-6, 23, and 32-35 are withdrawn in view of Applicant's amendments to the claims. The objections to Claims 36-38 are withdrawn in view of Applicant's cancellation of the claims.

8. The 35 U.S.C. § 112, second paragraph, rejection of Claim 10 is withdrawn in view of Applicant's amendments to the claim.

9. The 35 U.S.C. § 101 rejections of Claims 1-21 and 32-35 are withdrawn in view of Applicant's amendments to the claims. The 35 U.S.C. § 101 rejections of Claims 36-38 are withdrawn in view of Applicant's cancellation of the claims. However, the 35 U.S.C. § 101 rejections of Claims 26-31 are maintained in view of Applicant's arguments and amendments to the claims and further explained below.

Response to Amendment

Claim Objections

10. **Claims 26-31** are objected to because of the following informalities:

- **Claim 26** recites the limitations "objects of a second software subsystem" and "objects of the second software subsystem." Applicant is advised to change these limitations to read "software objects of a second software subsystem" and "software objects of the second software subsystem," respectively, for the purpose of keeping the claim language consistent throughout the claims.
- **Claims 27-31** depend on Claim 26 and, therefore, suffer the same deficiency as Claim 26.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

11. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

12. **Claims 26-31** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claim 26 recites the limitation of generating a compatibility check report. The subject matter is not properly described in the application as filed, since the originally-filed specification only discloses the changes monitor accessing the compatible changes database after a change has been detected *(see Page 7, Paragraph [0024])*. The specification lacks disclosure on the generation of a compatibility check report. Because the specification does not adequately support the claimed subject matter, it would not reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claims 27-31 depend on Claim 26 and, therefore, suffer the same deficiency as Claim 26.

Claim Rejections - 35 USC § 101

13. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

14. **Claims 26-31** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 26-31 are directed to apparatus. However, the recited components of the apparatus appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. Therefore, these claim limitations can be reasonably interpreted as

computer program modules—software *per se*. Furthermore, the specification discloses that many of the features and techniques may be implemented in software *(see Page 13, Paragraph [0040])*. The claims are directed to functional descriptive material *per se*, and hence non-statutory.

The claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claim Rejections - 35 USC § 102

15. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

16. **Claims 1-4, 6-20, 22-27, 29-34, 36, and 37** are rejected under 35 U.S.C. 102(e) as being anticipated by **Carter et al. (US 6,519,767)**.

As per **Claim 1**, Carter et al. disclose:

- automatically detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem *(see Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."; Column 14: 9-16, "... version compatibility analyzer 70 further compares the types of the classes and public members (including the member's parameters and return values). If the types of any of these identically named classes or public members do not match, new object server 64 is considered to have changed the class or public member over existing object server 66.")*;

- determining whether the change is compatible with the software objects of the second software subsystem *(see Column 14: 17-23, "... when new object server 64 is determined to have removed or changed types of any class or public member over existing object server 66 from the comparison step in 161, new object server 64 is determined by version compatibility analyzer 70 to be incompatible with existing object server 66.")*; and

- implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem *(see Column 8: 21-25, "When no version incompatible differences are detected by version compatibility analyzer 70, compatible object server generator 72 generates the interface related information in process 78 so as to be version compatible with existing object server 66."; Column 14: 33-38, "when new object server 64 is determined from the step 161 comparison to have added any class or any public member to a class over existing object server 66, new object server 64 is determined by version compatibility analyzer 70 to be version compatible with existing object server 66. In this situation, new object server 64 can be built to support each interface of existing object server 66 ...")*;

- otherwise, rejecting the introduced change and generating an error notification *(see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message.")*.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Carter et al. further disclose:

- wherein determining whether the change is compatible further comprises determining whether the change is predefined as compatible *(see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class.").*

As per **Claim 3**, the rejection of **Claim 2** is incorporated; and Carter et al. further disclose:

- allowing the change if the change is predefined as compatible *(see Column 13: 10-13, "... edits which do not change any interface supported by existing object server 66 can be made to source data 60 without making new object server 64 incompatible.").*

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and Carter et al. further disclose:

- issuing a message that the change is not allowed if the change is not predefined as compatible *(see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message.").*

As per **Claim 6**, Carter et al. disclose:

- identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen *(see Column 8: 30-37, "In the existing version object server 92, clock object 94 has three members 96-98 (FIG. 3A), respectively named "SetTime,"*

"GetHour," and "GetMinute," as well as properties 100 which include integer values named "hour," and "minute." In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."); and

- detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change *(see Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."; Column 14: 9-16, "... version compatibility analyzer 70 further compares the types of the classes and public members (including the member's parameters and return values). If the types of any of these identically named classes or public members do not match, new object server 64 is considered to have changed the class or public member over existing object server 66."),*

- determining with a compatibility check whether the change is compatible with a second software subsystem *(see Column 14: 17-23, "... when new object server 64 is determined to have removed or changed types of any class or public member over existing object server 66 from the comparison step in 161, new object server 64 is determined by version compatibility analyzer 70 to be incompatible with existing object server 66.")*; and

- issuing a notice indicating results of the compatibility check *(see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message.").*

As per **Claim 7**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein the subset of software objects declared frozen includes software objects of the first software subsystem that are used by the second software subsystem *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").*

As per **Claim 8**, the rejection of **Claim 7** is incorporated; and Carter et al. further disclose:

- wherein frozen software objects are classified to include released objects and restricted objects *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").*

As per **Claim 9**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein the released objects include software objects that are used by the second software subsystem without restrictions *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").*

As per **Claim 10**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein the restricted objects include software objects that are used by software objects of the second software subsystem, the software objects being fewer in number than a threshold *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").*

As per **Claim 11**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the

restricted object *(see Column 8: 55-59, "In this vtable, ITime$_1$ interface 110 includes pointers to members 96-98 of clock object 94. The members 96-98 can then be accessed by code in client application 101 based on their particular offset within the vtable.").*

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein classification of the frozen software objects is based on a number of times a frozen software object is used by the second software subsystem *(see Column 9: 1-6, "AddRef and Release functions 125-126 include code for OLE 2 to track the number of references from client applications currently using an instance of clock object so that memory allocated to such instance can be released when the instance is no longer in use.").*

As per **Claim 13**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object is a function module *(see Column 7: 13-16, "Integrated development environment 50 further includes a source editor 56 which provides conventional editing tools for a user to enter and modify programs, including programs which are to be compiled into object servers.").*

As per **Claim 14**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object is a data structure *(see Column 7: 27-31, "Source data 60 of the program can be stored by source editor 56 in the form of a data stream representation of the programming language statements, or more preferably a data stream consisting of a tokenized representation of the programming language statements.").*

As per **Claim 15**, the rejection of **Claim 13** is incorporated; and Carter et al. further disclose:

- wherein the software object includes an environment of the function module *(see Column 7: 13-16, "Integrated development environment 50 further includes a source editor 56 which provides conventional editing tools for a user to enter and modify programs, including programs which are to be compiled into object servers.").*

As per **Claim 16**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes a class and an environment of the class *(see Column 7: 25-27, "In the Visual Basic language system, each project can include one or more class modules which each define an object.").*

As per **Claim 17**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes an interface and an environment of the interface *(see Column 7: 25-27, "In the Visual Basic language system, each project can include one or more class modules which each define an object.").*

As per **Claim 18**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes a program and an environment of the program *(see Column 7: 23-25, "... each program is entered by the user in the form of a "project" in the Visual Basic language system.").*

As per **Claim 19**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein the detecting the change comprises automatically monitoring development of software code *(see Figure 6; Column 13: 36-46, "In process 76, version compatibility analyzer 70 determines version compatibility with existing object server 66 by comparing source data 60 to type information stored in existing object server's type library 150.").*

As per **Claim 20**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein the determining whether the change is compatible comprises determining whether there is a predefined declaration of compatibility of the change *(see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version*

compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class.").

As per **Claim 22,** Carter et al. disclose:

- performing a global compatibility check of software objects of a first software subsystem by determining whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check, wherein the introduced changes were introduced without obtaining prior approval *(see Figure 6; Column 12: 62-67, "FIG. 6 shows version compatibility analysis process 76 (FIG. 6) comprising steps 160-167 which is utilized by compiler 52 (FIG. 2) in version compatibility analyzer 70 (FIG. 2) to determine whether new object server 64 (FIG. 2) can be version compatible with existing object server 66 (FIG. 2).")*;

- identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system *(see Figure 6: 164 and 165; Column 7: 60-65, "The user also can select with user input 58 to have compiler 52 compile source data 60 utilizing the process according to the present invention for automatically building a version compatible object server, or in the conventional manner (without the version compatible object server building process)."; Column 8: 3-8, "If no file name is specified in the Compatible Object Application field (such as when compiling the first version of existing object server 66), compiler 52 compiles source data 60 conventionally into an object server without the automatic version compatible object server*

building process."; Column 14: 33-38, "As indicated at steps 164 and 165, when new object server 64 is determined from the step 161 comparison to have added any class or any public member to a class over existing object server 66, new object server 64 is determined by version compatibility analyzer 70 to be version compatible with existing object server 66."); and

- issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change *(see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message.")*.

As per **Claim 23**, the rejection of **Claim 22** is incorporated; and Carter et al. further disclose:

- wherein the performing a global compatibility check comprises comparing a current version of software code with a version of the software code at the time of a last global compatibility check *(see Column 12: 62-67, "FIG. 6 shows version compatibility analysis process 76 (FIG. 6) comprising steps 160-167 which is utilized by compiler 52 (FIG. 2) in version compatibility analyzer 70 (FIG. 2) to determine whether new object server 64 (FIG. 2) can be version compatible with existing object server 66 (FIG. 2).")*.

As per **Claim 24**, the rejection of **Claim 22** is incorporated; and Carter et al. further disclose:

- wherein the subset of the software objects includes frozen software objects *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding*

a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").

As per **Claim 25**, the rejection of **Claim 24** is incorporated; and Carter et al. further disclose:

- wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem *(see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99.").*

As per **Claim 26**, Carter et al. disclose:

- a changes monitor to automatically detect a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem, to perform a compatibility check to determine whether the change is compatible with the software objects of the second software subsystem, and to generate a compatibility check report that indicates whether the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem *(see Figure 2: 52; Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to*

detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."; Column 8: 21-25, "When no version incompatible differences are detected by version compatibility analyzer 70, compatible object server generator 72 generates the interface related information in process 78 so as to be version compatible with existing object server 66."; Column 14: 17-23, "... when new object server 64 is determined to have removed or changed types of any class or public member over existing object server 66 from the comparison step in 161, new object server 64 is determined by version compatibility analyzer 70 to be incompatible with existing object server 66." and 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message."); and

- an error notification module to notify a software developer introducing the change into the software object of the first software subsystem of a not allowed change if the change is incompatible, and otherwise indicating compatibility *(see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message." and 42-46, "... version compatibility analyzer 70 concludes that new object server 64 is version identical to existing object server 66.").*

As per **Claim 27**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the changes monitor to determine whether the change is compatible comprises the changes monitor to determine whether there is a predefined declaration of

compatibility of the change *(see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class.").*

As per **Claim 29**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- a master system including the changes monitor to detect a change introduced into a software object of a first software subsystem from a plurality of software subsystems *(see Figure 2: 50; Column 7: 1-5, "FIG. 2 is a block diagram of a programming language system or integrated development environment 50 which provides a compiler 52 for building version compatible object servers according to a preferred embodiment of the invention.").*

As per **Claim 30**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the first software subsystem is located at a server *(see Column 6: 16-18, "Referring to FIG. 1, an operating environment for the preferred embodiment of the present invention is a computer system 20 ...").*

As per **Claim 31**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the second software subsystem is located at a client *(see Column 6: 16-18, "Referring to FIG. 1, an operating environment for the preferred embodiment of the present invention is a computer system 20 ... ").*

Claims 32-34 are article of manufacture claims corresponding to the method claims above (Claims 1, 2, and 4) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1, 2, and 4.

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. **Claims 5, 21, 28, 35, and 38** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Carter et al. (US 6,519,767)** in view of **Hiller et al. (US 6,658,659)**.

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Carter et al. do not disclose:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller et al. disclose:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller et al. – Column 13: 6-12)*.

As per **Claim 21**, the rejection of **Claim 7** is incorporated; however, Carter et al. do not disclose:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

Hiller et al. disclose:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller et al. – Column 13: 6-12).*

As per **Claim 28**, the rejection of **Claim 26** is incorporated; however, Carter et al. do not disclose:

- wherein the change monitor to determine whether the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller et al. disclose:

- wherein the change monitor to determine whether the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to

include wherein the change monitor to determine whether the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller et al. – Column 13: 6-12).*

Claim 35 is rejected for the same reason set forth in the rejection of Claim 5.

Response to Arguments

19. Applicant's arguments filed on May 17, 2007 have been fully considered, but they are not persuasive.

In the remarks, Applicant argues that:

a) The Office Action refers to the claims as software per se, because the claims can reasonably be interpreted as including software elements. However, the Specification at paragraph [0040] explicitly states that "hardwired circuitry of firmware may be used in place of software, or in combination with software, to implement the features described herein. Thus, the invention is not limited to any specific combination of hardware circuitry and software ...." Applicants respectfully submit that just as a reasonable interpretation of the claims could infer the use of software elements, the Specification is clear to suggest that hardware circuits could also be included. Thus, Applicants submit that the Office Action is eliminating a possible

interpretation of the claims, supported by the Specification, to reject the claims. Per MPEP § 2111, "During patent examination, the pending claims must be given their broadest reasonable interpretation ...." Citations omitted. Applicants submit that by eliminating a possible interpretation described in the Specification, the Office Action is giving a narrowing interpretation to the claims, in contradiction of the principle recited in MPEP § 2111.

Examiner's response:

a) Examiner disagrees with Applicant's assertion that the Office action is giving a narrowing interpretation to the claims. The Examiner maintains that the 35 U.S.C. § 101 rejections of Claims 26-31 are consistent with the Office's current policies regarding non-statutory subject matter. Since the originally-filed specification discloses that many of the features and techniques may be implemented in software *(see Page 13, Paragraph [0040])*, the Examiner has reasons to believe that the claimed apparatus can be reasonably interpreted, under the broadest reasonable treatment, as computer program modules—software *per se*, based on intrinsic evidence of such. Although the cited paragraph discloses hardware elements that may be used to implement the features, the claims are interpreted as software *per se* under one possible interpretation supported by the disclosed evidence of software. Furthermore, the apparatus appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101, and therefore, suggestive of the apparatus as being directed to functional descriptive material *per se*, and hence non-statutory. See 35 U.S.C. § 101 rejections of Claims 26-31 above.

In the remarks, Applicant argues that:

b) Specifically, "In the preferred embodiment, version compatibility analyzer 70 performs this comparison by comparing the type information of each class in new object server 64 to the type information of [an] identically named class in existing object server's type library 150." Applicants note the significant absence of any detecting of a change to a software object, in contrast to what is recited in Applicants' claims. As appears from the cited reference, Carter fails to consider changes at all to a software object, and is rather concerned with changes to interfaces of object servers, which is not the same thing.

Examiner's response:

b) Examiner disagrees with Applicant's assertion that Carter et al. fail to consider changes at all to a software object. Carter et al. clearly disclose changes regarding a software object *(see Figure 2: 64 and 66; Column 7: 34-43, "Compiler 52 implements a process ... for automatically building a new object server 64 from source data 60 so as to be version compatible with an existing object server 66." and "... source data 60 is compiled through a sequence of compilation states to an executable state." and "... compile and generate an executable file (e.g. a file with the ".exe" extension, such as "Objserv.exe") for the object server which can be saved in secondary storage 40 for later execution."; Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."; Column 14: 9-16, "... version compatibility analyzer 70 further compares the types of the classes*

and public members (including the member's parameters and return values). If the types of any of these identically named classes or public members do not match, new object server 64 is considered to have changed the class or public member over existing object server 66.").

Note that Elements 64, "NEW OBJSERV.EXE" and 66, "EXISTING OBJSERV.EXE" of Figure 2 clearly indicate that the "object server" is a software object. The object server executable file is generated by a compiler and can be stored for later execution. Furthermore, Applicant has submitted in the originally-filed specification that a software object includes, but not limited to, function modules, programs, data objects, classes, class components, interfaces, attributes, etc. *(see Page 4, Paragraph [0016]).* An executable file clearly fits within the scope of this definition.

Conclusion

20. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

WEI ZHEN
SUPERVISORY PATENT EXAMINER

QC / QC
May 24, 2007